|  |  |  |
| --- | --- | --- |
| **File Database** | | |
| * S3 instance * One for each user | | |
| Store files |  | * Using AWS S3 which is optimised for file storage * S3 offers file encryption for all objects in the bucket |
| Prevent access from any other instances or applications except for the Cloud backend | * Security feature to prevent other instances from accessing the files | * Ingress/Egress rules? |
| **General Database** | | |
| * RDS instance (?) * Relational database for general purposes | | |
| Store usernames, emails and password hashes | * Account management for the users * Only accessible by the cloud backend for security |  |
| Actions log | * Log all user actions so that the user can view them in the future | * Audit table |
| **Cloud backend** | | |
| *file-manager* | | |
| * Flask server   + Boto3 (communicate to S3)   + Zappa (deployment tool for Lambda) * Python * ~~EC2~~ Lambda instance | | |
| Endpoint to upload file | * ~~Send file over SFTP~~ * Pre-signed url to upload to will be returned | * POST |
| Endpoint to download file | * ~~Send file over SFTP~~ * Pre-signed url for the given file will be returned | * GET |
| Endpoint to download the names of files currently stored | * Basic HTTP payload | * GET |
| Endpoint to retrieve user logs | * Basic HTTP payload | * GET |
| **Local backend** | | |
| *user-manager* | | |
| * Flask server * Python app * PyInstaller * Should launch automatically with frontend (Docker!) – can package with Electron app maybe? * <https://www.youtube.com/watch?v=ty-n33mHwC4> * <https://stackoverflow.com/questions/51455898/how-to-package-an-electron-app-and-flask-server-into-one-executable> * <https://advancedweb.hu/are-s3-signed-urls-secure/> * https://blog.miguelgrinberg.com/post/json-web-tokens-with-public-key-signatures | | |
| Encrypt file before sending | * Un-encrypted file should never leave local | * AES encryption (?) * PyCrypto or PyCryptodome modules |
| Decrypt file on download | * User to view decrypted version | * Key to be generated with the user’s password and should be left on the local   + Storage to be decided – split up the key? |
| Checks password against known password dictionaries | * Encourages users to use a unique pass­­­­word according to the latest NIST standards | * SkullSecurity has several of them available - <https://wiki.skullsecurity.org/Passwords> |
| Sends a code via email for setup and **all** logins | * Verification for login |  |
| ~~Sends file to cloud via SFTP~~ | * ~~Secure File Transport Protocol~~ | * ~~Pysftp module~~ |
| Scan through files | * Checking for any sensitive information |  |
| **Frontend application** | | |
| *user-interface* | | |
| * Electron * Javascript/React * Electron Packager | | |
| Drag and drop file | * Should allow users to drag and drop files to be uploaded * Box should also be selectable, and the file browser will be brought up | * https://react-dnd.github.io/react-dnd/about |
| Check what files have been stored |  |  |
| Log in and log out | * Token functionality? | * JWT |